ICALEPCS'13

14th International Conference on Accelerator & Large Experimental Physics Control Systems
October 6-11, 2013, San Francisco, California

PLUGIN-BASED ANALYSIS FRAMEWORK FOR LHC POST-MORTEM ANALYSIS

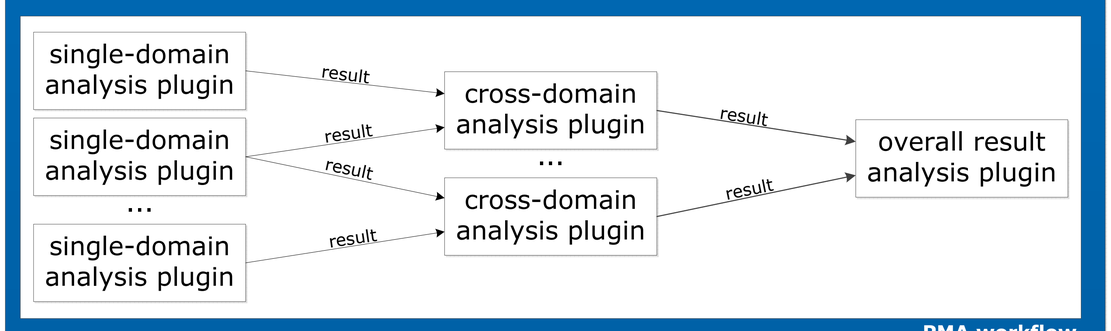
R. Gorbonosov, G. Kruk, M. Zerlauth, V. Baggiolini, CERN, Geneva, Switzerland

Abstract

Plugin-based software architectures [1] are extensible, enforce modularity and allow several teams to work in parallel. But they have certain technical and organizational challenges. We gained our experience when developing the Post-Mortem Analysis (PMA) system, which is a mission-critical system for the Large Hadron Collider (LHC). We used a plugin-based architecture with a general-purpose analysis engine, for which physicists and equipment experts code plugins containing the analysis algorithms. We have over 45 analysis plugins developed by a dozen of domain experts. This paper focuses on the design challenges we faced in order to mitigate the risks of executing third-party code.

1. Background

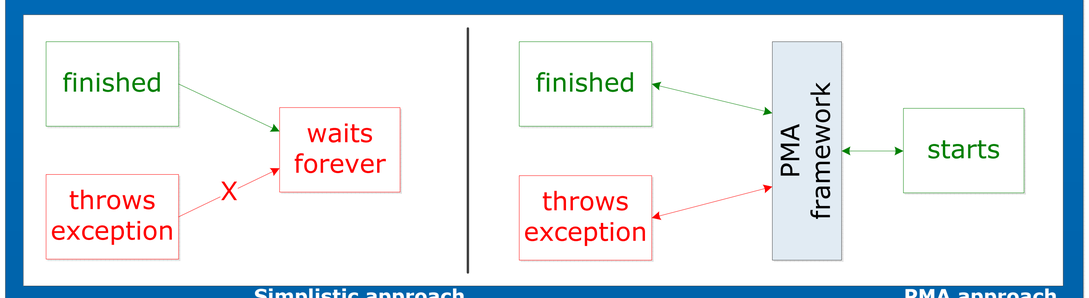
The Post-Mortem Analysis (PMA) performs an exhaustive analysis of the behavior and state of the key LHC components. Analysis implementation requires detailed domain knowledge. Therefore domain experts contribute to the overall analysis writing analysis plugins executed by general-purpose engine. Domain experts are not professional programmers and prone to make mistakes. Plugins are executed in certain order: subsequent plugins consume the output of previous plugins.



PMA workflow

2. Plugin execution

A simplistic approach would allow plugins to execute autonomously notifying each other about produced results. Seeming natural, this approach compromises the overall analysis execution in case a plugin fails with exception. It also forces the domain experts to keep track of incoming data and to send notifications.

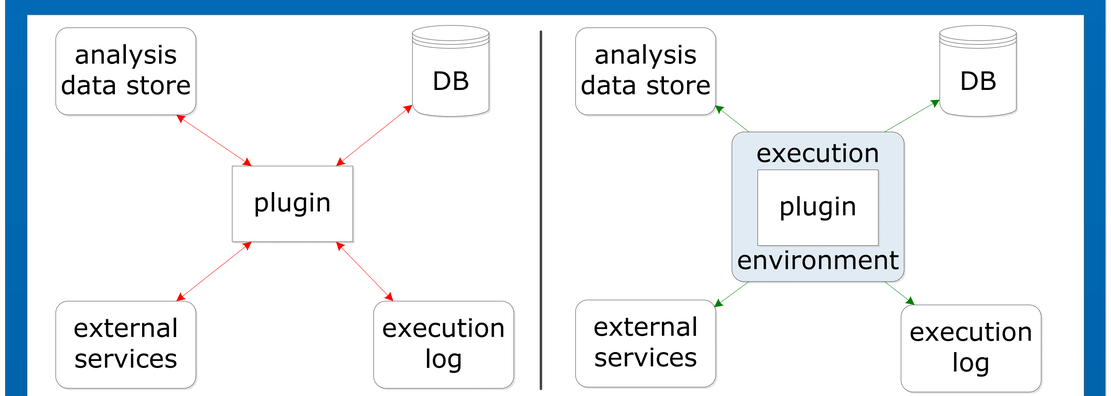


Simplistic approach PMA approach

In PMA the framework controls the plugin execution entirely. This guarantees the execution of all the analysis logic and simplifies the code of plugins.

3. Plugin misbehavior

There are many ways a plugin can fail: it can block, access resources or services too often, produce an enormous amount of data. A simplistic approach would let plugin access the resources and services directly. Seeming the simplest, this approach compromises the overall analysis execution. A plugin overloading services used by other plugins can potentially bring the services down preventing other plugins from finishing successfully.



Simplistic approach PMA approach

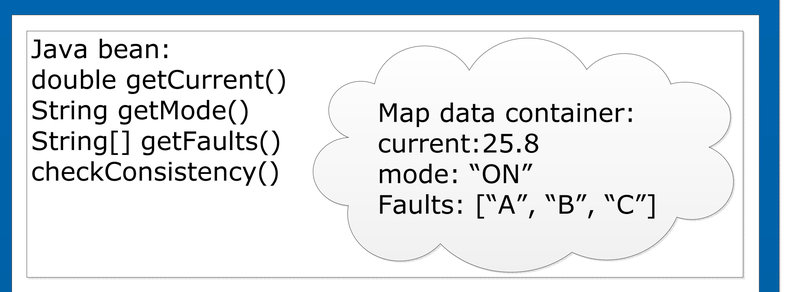
In PMA each plugin is executed in a special environment which provides access to resources and services via proxies. The proxies allow the framework to intervene if plugin misbehavior is detected.

4. Inter-plugin communication

Inter-plugin communication requires the definition of clear contract between communicating plugins:
*** data container and format (XML, JSON, hash maps, etc.)**
*** data content (data items and their representation)**

A simplistic approach would allow the plugins to produce data in their own formats. Seeming flexible this approach is chaotic and introduces complexity for the consuming plugins. In PMA we have chosen maps with key-value pairs as the standard data format.

The standard format provides no guarantee for the data consistency. Neither it allows to find out at development time which data items are expected and how data items are represented. In PMA map data containers are wrapped into a data-specific Java beans [2]. Java beans allow the data consistency check at runtime. In addition, the developer of a consuming plugin gets the full power of compilation check and IDE code-completion.



Java bean wrapper over map data container

Conclusions

The PMA framework has been used operationally for several years and proved to be very extensible, flexible and reliable. At CERN there are currently 4 mission-critical LHC applications based on the PMA framework: Global PMA [3], Injection Quality Check [4], External Post-Operational Check of LHC beam-dump system [5] and Powering Event Analysis. In total there are over 45 analysis plugins developed by a dozen of domain experts. Such a broad adoption would have never been possible without a plugin-oriented architecture and the design decisions described in this article.

REFERENCES

[1] http://en.wikipedia.org/wiki/Plugins
[2]http://www.oracle.com/technetwork/java/javase/documentation/spec-136004.html
[3] M. Zerlauth et al, "The LHC Post Mortem Analysis Framework", TUP021, ICALEPCS 2009, Kobe, Japan
[4] L. N. Drosdal et al, "Automatic Injection Quality Checks for the LHC", WEPMU011, ICALECS 2011, Grenoble, France
[5] N. Magnin et al, "External Post-Operational Checks for the LHC Beam Dumping System", WEPMU023, ICALECS 2011, Grenoble, France